Refine Search

Search Results -

Term	Documents
(37 AND 2).USPT.	4
(L2 AND L37).USPT.	4

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L39

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, May 31, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT; PLUR=YES; OP=ADJ		
L39	l2 and L37	4	L39
L38	l32 and L37	5	L38
L37	l21 and L36	25	L37
L36	l25 and L35	28	L36
L35	workload near4 manag$	278	L35
L34	l31 and l32 and L33	47	L34
L33	interval$1	496819	L33
L32	poll$	150752	L32
L31	l29 and L30	60	L31
L30	workload$1	5576	L30
L29	l27 and L28	393	L29
L28	queue$1	29282	L28
L27	l12 and L26	1669	L27

L26	l21 and l25	8303	L26
L25	sampling	134081	L25
L24	l20 and l21	13	L24
L23	l20 and L22	0	L23
L22	updat$ near3 workload$1	34	L22
L21	resource$1	87378	L21
L20	l18 and L19	27	L20
L19	queue near4 interval$1	510	L19
L18	sampling near5 interval$1	13373	L18
L17	queue interval$1	15	L17
L16	l3 and L15	8	L16
L15	l10 and l14	133	L15
L14	poll$ near5 resource$1	1260	L14
L13	l11 and L12	23	L13
L12	normaliz$	66074	L12
L11	l2 and l3 and l7 and l10	45	L11
L10	resource$1 near4 allocat$	9111	L10
L9	reaource$1 near4 allocat$	0	L9
L8	l4 and L7	15	L8
L7	process$ near4 (interval$1 or duration$1)	27811	L7
L6	l4 and L5	4	L6
L5	queue length or queue-length	984	L5
L4	l1 and l2 and L3	64	L4
L3	workload$	5618	L3
L2	probabil$ or stochastic or rendom$ or normaliz$	149666	L2
L1	resource near3 select$	2989	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Term	Documents
(37 AND 2).USPT.	4
(L2 AND L37).USPT.	4

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L39

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, May 31, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L39	l2 and L37	4	L39
L38	l32 and L37	5	L38
L37	l21 and L36	25	L37
L36	l25 and L35	28	L36
L35	workload near4 manag$	278	L35
L34	l31 and l32 and L33	47	L34
L33	interval$1	496819	L33
L32	poll$	150752	L32
L31	l29 and L30	60	L31
L30	workload$1	5576	L30
L29	l27 and L28	393	L29
L28	queue$1	29282	L28
L27	l12 and L26	1669	L27

L26	l21 and l25	8303	L26
L25	sampling	134081	L25
L24	l20 and l21	13	L24
L23	l20 and L22	0	L23
L22	updat$ near3 workload$1	34	L22
L21	resource$1	87378	L21
L20	l18 and L19	27	L20
L19	queue near4 interval$1	510	L19
L18	sampling near5 interval$1	13373	L18
L17	queue interval$1	15	L17
L16	l3 and L15	8	L16
L15	l10 and l14	133	L15
L14	poll$ near5 resource$1	1260	L14
L13	l11 and L12	23	L13
L12	normaliz$	66074	L12
L11	l2 and l3 and l7 and l10	45	L11
L10	resource$1 near4 allocat$	9111	L10
L9	reaource$1 near4 allocat$	0	L9
L8	l4 and L7	15	L8
L7	process$ near4 (interval$1 or duration$1)	27811	L7
L6	l4 and L5	4	L6
L5	queue length or queue-length	984	L5
L4	l1 and l2 and L3	64	L4
L3	workload$	5618	L3
L2	probabil$ or stochastic or rendom$ or normaliz$	149666	L2
L1	resource near3 select$	2989	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

☐ Generate Collection Print

L39: Entry 2 of 4 File: USPT Jun 3, 2003

DOCUMENT-IDENTIFIER: US 6574587 B2
TITLE: System and method for extracting and forecasting computing resource data such as CPU consumption using autoregressive methodology

Abstract Text (1):
A system and method for extracting and forecasting computing resource data such as workload consumption of mainframe computing resources using an autoregressive model. The system and method forecast mainframe central processing unit (CPU) consumption with ninety-five percent accuracy using historical performance data. The system and method also provide an upper ninety-five percent confidence level and a lower ninety-five percent confidence level. The system and method retrieve performance records from a computer platform in one second intervals, statistically collapses the one second performance data into fifteen minute performance data, statistically collapses the fifteen minute performance data into one week performance data, and generates a time series equivalent to collecting performance data at one week intervals. The system and method ensure that the resulting time series is statistically stationary, and applies an autoregressive construct to the time series to generate forecast of future CPU utilization, as well as to generate reports and graphs comparing actual vs. forecast CPU utilization. Because the system and method rely on electronically generated empirical historical computer performance data as an input, they provide a turnkey solution to CPU consumption forecasting that can be implemented easily by any system network manager.

Brief Summary Text (2):
The present invention relates to a computer platform, and in particular, to a system and method to forecast the performance of computing resources.

Brief Summary Text (4):
The computing resources of a large business represent a significant financial investment. When the business grows, resource managers must ensure that new resources are added as processing requirements increase. The fact that the growth and evolution of a computing platform is often rapid and irregular complicates management efforts. This is especially true for computing platforms common to banking institutions and telecommunications companies, for example, whose computing platforms typically include hundreds of geographically distributed computers.

Brief Summary Text (5):
To effectively manage the vast resources of a computing platform and to justify any requests for acquisition of new resources, managers need accurate forecasts of computing platform resource performance. However, conventional forecasting tools may not be adequate for use on computing platforms. For example, conventional sales performance forecasting tools, which use linear regression and multivariable regression to analyze data, commonly factor in such causal variables as the effect of holiday demand, advertising campaigns, price changes, etc. Similarly, pollution forecasting tools typically consider the causal effect of variations in traffic patterns. As such, using these tools to forecast computing platform resources may be problematical because causal parameters generally are difficult to establish and are unreliable.

Brief Summary Text (7):

The limitations of established forecasting tools are particularly troublesome when forecasting resources in computing platforms that are expanding or are already re-engineered. These computing platforms need a forecasting system and method that deal appropriately with new data as well as unneeded data. Moreover, these computing platforms need a forecasting system and method that augment causal-based forecasting tools to provide accurate and reliable forecasts.

Brief Summary Text (9):
Presented herein is a system and method to forecast computing platform resource performance that overcomes the limitations associated with conventional forecasting tools. An embodiment applies an autoregressive model to electronically generated empirical data to produce accurate and reliable computing platform resource performance forecasts. An embodiment of the present invention also statistically collapses large amounts of data, eliminates unneeded data, and recursively processes new data. The forecasts are compared to actual performance data, which may be graphically displayed or printed. A specific type of data is not important for the present invention, and those skilled in the art will understand that a wide variety of data may be used in the present invention. For example, the present invention contemplates any data that may be collected and verified over time. These data include, for example, Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc.

Brief Summary Text (11):
The computing platform includes at least one resource whose performance is forecast. In one embodiment, the computing platform resource may be a central processing unit (CPU). In another embodiment, the computing platform resource may be a memory storage unit. In other embodiments, the computing platform resource may be a printer, a disk, or a disk drive unit. A specific computing platform resource is not important for the present invention, and those skilled in the art will understand that a number of resources may be used in the present invention.

Brief Summary Text (12):
Each resource includes at least one aspect. The aspect may be a performance metric. The performance metric may be resource utilization. "Utilization" is defined generally herein as the percentage that a particular computing platform resource is kept busy. Utilization is often termed "consumption."

Brief Summary Text (13):
In another embodiment, the performance metric may be resource efficiency or resource redundancy. "Efficiency" is defined generally herein as the measure of the useful portion of the total work performed by the resource. "Redundancy" is defined generally herein as the measure of the increase in the workload of a particular resource. Of course, those skilled in the art will appreciate that a particular performance metric is not required by the present invention. Instead, a number of performance metrics may be used.

Brief Summary Text (14):
In one embodiment, the computing platform includes a resource manager. The resource manager collects performance data from its associated resource. The performance data is associated with a performance metric. In one embodiment, the resource manager collects performance data representing a CPU utilization performance metric.

Brief Summary Text (15):
The resource manager collects the performance data in regular intervals. In one embodiment, regular intervals include one-second intervals, for example. That is, in this embodiment, the resource manager collects performance data from its associated computer(s) every second. The interval size in which performance data is

collected may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (17):
A first statistical collapser generates a first time series representing a performance metric as though its associated performance data had been collected at a first interval. The first time series includes a first set of time points. In one embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected in fifteen minute intervals. Accordingly, the time series includes four time points for each hour. In another embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected hourly. Accordingly, the time series includes one time point for each hour. It will be understood by persons skilled in the relevant art that the present invention encompasses statistical collapsers that generate time series representing performance metrics as though their associated performance data had been collected at any of a variety of suitable intervals. The interval size and corresponding number of time points generated by the first statistical collapser may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (20):
The computing platform also includes a second statistical collapser. The second statistical collapser statistically collapses the first time series, producing a second time series. The second time series includes a second set of time points. In one embodiment, the second statistical collapser statistically collapses the fifteen minute time series into a one-week time series. That is, the second statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected weekly. Accordingly, the time series includes approximately four time points for each month. In another embodiment, the second statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected daily. The corresponding time series includes approximately thirty time points for each month. It will be understood by persons skilled in the relevant art that the second statistical collapser may generate time series representing a performance metric as though its performance data had been collected at any of a variety of suitable intervals. As described above with reference to the first statistical collapser, the interval size and corresponding number of time points generated by the second statistical collapser may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (23):
One feature of the present invention is an autoregressive modeling tool, which is applied to the converted time series to forecast a particular aspect of the computing platform. The autoregressive modeling tool is chosen by calculating autocorrelation, inverse autocorrelation, and partial autocorrelation functions, and by comparing these functions to theoretical correlation functions of several autoregressive constructs. In particular, one embodiment applies a first order mixed autoregressive construct, such as an autoregressive moving average (ARMA) construct, to the differenced time series. Another embodiment applies an autoregressive integrated moving average (ARIMA) construct to the differenced time series. In the embodiment where the performance metric is resource utilization and the resource is a CPU, the resulting autoregressive modeling tool reliably forecasts CPU consumption with a ninety-five percent accuracy, provides an upper ninety-five percent confidence level, and provides a lower ninety-five percent confidence level. Conventional systems and methods that rely on linear regression or multivariable regression techniques may carry a lower confidence level.

Brief Summary Text (24):

Another feature of the present invention is that it uses empirical data as inputs to the autoregressive modeling tool. Using empirical data rather than causal variables provides more accurate forecasts. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit, the empirical data is actual historical performance data, including logical CPU utilization information as well as physical CPU utilization information. Moreover, the system and method generate recursive forecasts whereby actual future performance data is fed back into the autoregressive modeling tool to calibrate the autoregressive modeling tool.

Brief Summary Text (26):
In one embodiment, the results processor may be a graphical display unit, such as a computer display screen. In another embodiment, the results processor may be a textual display unit, such as a printer. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit, the results processor produces reports and graphical representations of comparisons of actual CPU utilization with CPU utilization forecasts.

Detailed Description Text (2):
A computer platform, and in particular, a system and method for forecasting computer platform resource performance is described herein. In the following description, numerous specific details, such as specific statistical symbols and relationships, specific methods of analyzing and processing computer performance data, etc., are set forth in order to provide a full understanding of the present invention. One skilled in the relevant art, however, will readily recognize that the present invention can be practiced without one or more of the specific details, or with other methods, etc. In other instances, well-known structures or operations are not shown in detail in order to avoid obscuring the present invention.

Detailed Description Text (3):
For illustrative purposes, embodiments of the present invention are sometimes described with respect to a system and method for forecasting computer platform resource performance. It should be understood that the present invention is not limited to these embodiments. Instead, the present invention contemplates any data that may be collected and verified over time. These data may include, for example, Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc.

Detailed Description Text (9):
The computing platform 100 includes at least one resource. In one embodiment, the computing platform resource may be a central processing unit (CPU). In another embodiment, the computing platform resource may be a memory storage unit. In other embodiments, the computing platform resource may be a printer, a disk, or a disk drive unit. While a specific computing platform resource is not important for the present invention, those skilled in the art will understand that any number of resources can be used in the present invention.

Detailed Description Text (10):
Each resource includes at least one aspect. The aspect may be a performance metric. In one embodiment the performance metric may be resource utilization. Utilization is the measure of the percentage that a particular computing platform resource is kept busy, and is sometimes termed consumption. In another embodiment, the performance metric may be resource efficiency, which is defined as the measure of the useful portion of the total work performed by the resource. In another embodiment, the performance metric may be resource redundancy, which is defined as the measure of the increase in the workload of a particular resource. Of course, those skilled in the art will appreciate that a particular performance metric is not required by the present invention. Instead, the present invention supports any

of a number of performance metrics.

Detailed Description Text (11):
FIG. 2 is a more detailed block diagram of the computing platform 100 according to one embodiment. As illustrated, each computer 106 includes a resource manager 202. Each resource manager 202 collects performance data from its associated resource. The performance data is associated with a performance metric. According to one embodiment, the resource manager 202 is a resource management facility (RMF) available with the multiple virtual storage (MVS) operating system that is running on the IBM mainframe computer as noted above or an equivalent mainframe computer available from Amdahl and Hitachi Data Systems. According to this embodiment, the resource manager 202 extracts historical performance data from a processor resource/systems manager (PR/SM) (not shown) of the computer 106. This historical computer performance data represents the CPU utilization and is equivalent to performance metering data obtained by real-time monitors. Thus, the CPU utilization information collected by the resource manager 202 are CPU utilization records that contain CPU activity measurements.

Detailed Description Text (12):
The resource manager 202 collects the performance data from the computer 106 at regular intervals. According to an exemplary embodiment, the regular intervals are one-second intervals. That is, according to the exemplary embodiment, the resource manager collects CPU workload performance data every second from computer 106. In this way, the resource manager 202 provides the percent busy for each computer 106 each second in time. The interval size in which performance data is collected may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc.

Detailed Description Text (13):
Because the computers 106 typically are maintained by large entities, the amount of data collected usually is quite large. Consequently, the data must be reduced to a manageable level. Statistically collapsing the one-second records generated by the resource manager 202 serves this purpose. The computing platform 100 thus also includes a plurality of statistical collapsers that statistically collapse the performance data into time series representing a performance metric. A "time series" is defined herein generally as any ordered sequence of observations. Each observation represents a given point in time and is thus termed a "time point." A statistical collapser averages a series of time points and generates a time series representing a performance metric as though its associated performance data had been collected at a particular interval. The resulting time series contains a set of time points commensurate with the representative collection interval.

Detailed Description Text (14):
According to one embodiment, the computing platform 100 includes a statistical collapser 204 that statistically collapses the performance data collected by the resource manager 202 into a time series. The statistical collapser 204 generates a time series representing performance data as though it had been collected at fifteen-minute intervals. Accordingly, the time series would include four time points for each hour. In another embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected hourly. Accordingly, the time series would include one time point for each hour.

Detailed Description Text (15):
Thus, the statistical collapser 204 statistically collapses the CPU utilization records generated every second by the resource manager 202 into CPU utilization records representing fifteen-minute intervals. Nine hundred original CPU utilization records ([60 seconds/minute].times.[15 minutes]=900) are averaged to produce one collapsed time point. The statistical collapser 204 calculates the mean for all metering records collected by the resource manager 202, as described in

Stochastic

greater detail below. The statistical collapser 204 then determines the median for each mean at fifteen-minute intervals. The time series generated by the statistical collapser 204 thus consists of four data points (or time points) per hour representing the mean CPU utilization percentage. It will be understood by persons skilled in the relevant art that the present invention encompasses statistical collapsers that generate time series representing performance metrics as though its associated performance data had been collected at any of a variety of suitable intervals. The interval size and corresponding number of time points generated by the statistical collapser may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc.

Detailed Description Text (16):
A stochastic process, such as the time series representing the performance metric as though its performance data had been collected ~~at fifteen-~~minute intervals, may be represented by Z(.omega.,t). As used herein, a stochastic process is generally a family of time indexed random variables, Z(.omega.,t), where .omega. belongs to a sample space and t belongs to a time index set. ~~That is,~~ for a fixed time, t, Z (.omega.,t) is a random variable. For a given .omega., Z(.omega.,t), as a function of time, t, is called a sample function or realization. Thus, a time series is a realization or sample function from a certain stochastic process. Typically, however, the variable .omega. is suppressed, and the process is written Z(t) or Z.sub.t. The process is then called a real-valued process because it assumes only real values. The imaginary value, .omega., is not treated. Moreover, for any given real-valued process {Z(.omega.,t): t=0,.+-.1,.+-.2, . . . }, the mean function of the process is given by .mu..sub.t =e(Z.sub..function.t), which may be used by the statistical collapser 204 to calculate the mean for all metering records collected by the resource manager 202.

Detailed Description Text (17):
The computing platform also includes a database 206 that stores data. In one embodiment, the database 206 stores the time series representing a performance metric as though its associated performance data had been collected at fifteen-minute intervals. That is, after the resource manager 202 collects the performance data from the computers 106 and after the statistical collapser 204 generates the time series representing performance data collected at fifteen-minute intervals, the database 206 stores the time series.

Detailed Description Text (18):
The database 206, in one embodiment, is capable of storing at least sixty gigabytes of performance data and can process at least one record per second. For example, the database 206 stores thousands of mainframe computer performance statistical descriptors and resource utilization data. Although the database 206 is depicted as a single database, according to the exemplary embodiment, the database 206 may be a plurality of databases. A database suitable for implementing the database 206 is a MICS database available from Computer Associates located in Santa Clara, Calif., or an SAS IT Service Vision database available from SAS Institute located in Cary, N.C.

Detailed Description Text (22):
It must be noted that if a time series contains too few time points, the time series may not be representative of the particular data under analysis, including, but not limited to Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc. For example, if the time series in the above example embodiment contains too few time points, the time series may not be representative of actual resource performance. That is, peak usages (or spikes) may not be detected if too few time points are taken. Therefore, sampling intervals which exclude such peaks may inaccurately represent the resource utilization. Thus, the number of time

points in the time series may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc. To illustrate, suppose a network manager that is responsible for monitoring the behavior and effectiveness of the computing platform 100 resources monitors the performance and activities for each computer 106. The system network manager tracks the computing platform 100 resources performances by gathering the appropriate performance data from each component or network element in the computing platform 100. As described, performance metrics to be monitored include, but are not limited to, CPU consumption percentage, disk drive usage percentage, Internet traffic, users logged on to the Internet, network communication packet traffic, and users logged on to a particular server computer, for example.

Detailed Description Text (23):
Suppose further that the computing network 102 typically includes entities such as a configuration management team and a performance management team. The configuration management team would plan the computing platform 100's growth and modernization. Accordingly, weekly data points would be adequate for these network planning purposes. Daily data points may be more appropriate for use by the performance management team, however. This is because the performance management team would be concerned with maintaining the computing platform 100's error-free performance. Accordingly, the performance management team would be concerned about peak usages (or spikes) in resource consumption of the computing platform 100. Monitoring spikes in the computing platform 100 would facilitate load sharing, for example.

Detailed Description Text (24):
Referring to FIG. 3, one embodiment of the present invention generates accurate CPU utilization descriptors in the following manner. The resource manager 202 for the computer 106 collects the performance data 301 and provides it in the form of one-second metering records 302 to the statistical collapser 204. The statistical collapser 204 statistically collapses the one-second records 302 into fifteen-minute data, which is stored in a file of performance data 304 of the database 206. The data extractor 208 extracts the performance data 304 from the database 206 and provides it to the statistical collapser 210. The statistical collapser 210 statistically collapses the fifteen-minute data into one-week data.

Detailed Description Text (26):
Recall that a time series analogous to the one-week interval data resulted from collapsing the fifteen-minute interval data. According to the constraints of one embodiment, this time period must be statistically stationary. A statistically stationary time series is generally regarded as a time series which as a stochastic process, as defined above, is unchanged by a uniform increment in the time parameter defining the time series.

Detailed Description Text (27):
It must be noted that few time series are statistically stationary. The computing platform 100 thus includes a time series analyzer 212 to determine whether the time series generated by the statistical collapser 210 is statistically stationary. According to one embodiment, the time series analyzer 212 analyzes probability values for a plurality of X.sup.2 (chi-square) tests to make this determination. "Chi-square tests" as used herein generally define generalizations and extensions of a test for significant differences between a binomial population and a polynomial population, wherein each observation may fall into one of several classes and which furnishes a comparison among several samples rather than just between two samples. Such chi-square tests include a test of residuals, tests of hypotheses, tests of significance, tests of homogeneity, tests of association, goodness of fit tests, etc., as is known in the relevant art. In the embodiment where the resource is computer 106 and the performance metric is CPU utilization, the time series analyzer 212 determines whether there is a statistically significant correlation between a particular CPU utilization value and the value

for CPU utilization for the previous time period by reviewing correlation and covariance statistics. Tables 1-4 list chi-square values for a test for residuals for the computers 106a-106d, respectively. The column "DF" refers to the degrees of freedom of variation among a set of scores. In particular, column "DF" refers to the degrees of freedom of variation among a set of metering records for CPU utilization. To illustrate, suppose there is a set of ten scores. Statistically the degrees of freedom given by

Detailed Description Text (40):
II. Autoregressive Forecasting of Computing Resources

Detailed Description Text (41):
FIG. 4 depicts a flow chart of a collecting, collapsing, and regresssing process 400 suitable for use in one embodiment of the present invention. Task 402 starts the process 400, where control immediately passes to task 404. Task 404 extracts performance data from at least one of the computers 106. In one embodiment, the resource manager 202 extracts the performance data from its associated computer every second.

Detailed Description Text (45):
Task 412 applies the autoregressive modeling tool 216 to the time series to generate forecasts of the computing platform 100 resources. Task 412 also generates recursive forecasts whereby actual future performance data is fed back into the autoregressive modeling tool 216 to calibrate the system and method. Task 414 completes the process 400. This process provides a turnkey solution to CPU utilization forecasting that can be implemented easily by any system network manager.

Detailed Description Text (46):
Referring back to FIG. 1, the computing platform 100 may include a results processor 104. The results processor 104 generates graphical representations of performance data extracted from the computing platform 100. The results processor 104 generates information for use in written reports that document the results of the process 400. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit of the computers 106, the results processor 104 produces reports and graphical representations of comparisons of actual CPU utilization with CPU utilization forecasts.

Detailed Description Text (55):
These and other changes can be made to the invention in light of the above-detailed description. In general, in the following claims, the terms used should not be construed to limit the invention to the specific embodiments disclosed in the specification and claims, but should be construed to include all computer platforms that operate under the claims to provide a system and method for computing resource forecasting utilization.

CLAIMS:

1. In a computing platform having a plurality of resources, each resource includes at least one aspect, a method for forecasting at least one aspect of the plurality of resources to produce computing platform resource performance forecasts, the method comprising the steps of: collecting at intervals performance data associated with a performance metric from a computing platform resource; statistically collapsing the collected performance data associated with a performance metric at least once to produce a time series and storing the time series; determining whether the time series is statistically stationary using a plurality of chi-square tests; converting the time series to a statistically stationary time series when the time series is determined not to be statistically stationary; bypassing the converting when the time series is determined to be statistically stationary; adding a date/time stamp to the time series, including converting the time series

date/time stamp to a number of seconds equivalent to the value represented by the date/time stamp; applying an autoregressive modeling tool to the time series to produce computing platform resource performance forecasts; and outputting the resource performance forecasts in a format such that computing platform resources can be modified based on the data.

2. A system to forecast performance of at least one computing platform resource, comprising: a resource manager for collecting performance data associated with a performance metric from a computing platform resource; a first statistical collapser receiving input from the resource manager, the first statistical collapser collapsing the data received from the resource manager into a first time series; a second statistical collapser, the second statistical collapser collapsing the data received from the first statistical collapser into a second time series; a time series analyzer coupled to the second statistical collapser and configured to: determine whether the second time series is statistically stationary using a plurality of chi-square tests, convert the second time series to a statistically stationary time series when the second time series is determined not to be statistically stationary, and bypass the converting when the second time series is determined to be statistically stationary; a time point converter coupled to the time series analyzer, the time point converter being operable to add a date/time stamp to the second time series and wherein the time point converter converts the time series date/time stamp to a number of seconds equivalent to the value represented by the date/time stamp; an autoregressive modeling tool coupled to the time series analyzer, the autoregressive modeling tool producing computing platform resource performance forecasts; and a result processor coupled to the autoregressive modeling tool.

6. The system according to claim 2, wherein the performance metric represents utilization and the computing platform resource comprises a central processing unit.

First Hit Fwd Refs

☐ Generate Collection Print

L39: Entry 1 of 4 File: USPT Dec 2, 2003

DOCUMENT-IDENTIFIER: US 6658473 B1
TITLE: Method and apparatus for distributing load in a computer environment

Abstract Text (1):
The present invention provides a method and apparatus for distributing load in a multiple server computer environment. In one embodiment, a group manager process on each server periodically determines the server's capacity and load (i.e., utilization) with respect to multiple resources. The capacity and load information is broadcast to the other servers in the group, so that each server has a global view of every server's capacity and current load. When a given terminal authenticates to a server to start or resume one or more sessions, the group manager process of that server first determines whether one of the servers in the group already is hosting a session for that user. If that is the case, one embodiment of the present invention redirects the desktop unit to that server and the load-balancing strategy is not employed. Otherwise, for each resource and server, the proper load balancing strategies are performed to identify which server is best able to handle that particular session.

Brief Summary Text (6):
In this type of architecture, a large number of end users can connect to a limited number of servers. In addition, the limited number of servers are also interconnected, creating what is termed as a multiple server environment wherein any of the end user terminals could potentially connect to any of the servers. In multiple server environments it is common for the environment to be heterogeneous, in that each server has differing resource capabilities. In such complex multiple server environments, the load on the servers' resources often becomes unbalanced, meaning, for example, that one server is performing at essentially maximum capacity while another server is relatively unused. Overcoming this load imbalance, therefore, becomes an extremely important concern, if a high performance computing experience is to be provided.

Brief Summary Text (12):
In a multiple server environment, multiple sessions may be executing on each server. These sessions are initiated by multiple users accessing the DTUs. If one of these servers fails (e.g., loses power), each of the DTUs connected to it "fails over" to one of the surviving servers. Since the computational and memory resources allocated to the services requested by the DTUs are distributed across the group of servers, it is possible for resources to become unevenly allocated, thereby degrading performance on over-utilized servers while wasting resources on under-utilized servers. This is especially true in heterogeneous server configurations, where the carrying capacity of the servers (i.e., number and speed of processing units, amount of installed memory and available network bandwidth, for instance) is non-uniform. In addition, each session may demand differing quantities of resources adding to the non-uniformity of the resources allocated.

Brief Summary Text (13):
Furthermore, the presence of failures complicates the load distribution problem, because if a server hosting a large number of DTUs fails, all of the DTUs will attempt to fail over within a short time period. It is crucial in this situation not to unbalance the remaining servers by connecting failed over sessions to a

single server or an already overburdened server. Clearly, a more intelligent load balancing strategy is needed to achieve optimal resource allocation in this complex multiple server environment.

Brief Summary Text (15):
The present invention provides a method and apparatus for distributing load in a multiple server computer environment. In one embodiment, a group manager process on each server periodically determines the server's capacity and load (i.e., utilization) with respect to multiple resources. The capacity and load information is broadcast to the other servers in the group, so that each server has a global view of every server's capacity and current load.

Brief Summary Text (16):
When a user attempts to access a DTU, the user inserts an identifier into the DTU which contains a unique token. This identifier is a smart card, in one embodiment. Once the identifier is inserted, the DTU uses the token to attempt to establish communications with the servers to start or resume one or more sessions. When a given DTU successfully starts or resumes a given session, the group manager process of that server first determines whether one of the servers in the group already is hosting the session for that token. If that is the case, one embodiment redirects the DTU to that server and the load-balancing strategy is not employed. Otherwise, for each resource and server, the proper load balancing strategies are performed to identify which server is best able to handle that particular session.

Brief Summary Text (18):
In another embodiment, sessions are assigned to servers in a pseudo-random fashion, with the relative probability of selection of a server being weighted by its relative desirability. Pseudo-random selection is used primarily in fail over situations in which many sessions are being concurrently authenticated. In another embodiment, a hybrid strategy is used, which combines the use of the relative desirability strategy, and the use of pseudo-random strategy depending on the state of the server at that time. Thus, load balancing strategies result in a higher performance computing experience for the end user.

Detailed Description Text (3):
One or more embodiments of the invention may implement the mechanisms for improved resource utilization described in U.S. patent application Ser. No. 09/513,052, filed on Feb. 25, 2000, entitled "Method and Apparatus for Improving Utilization of a Resource on a Shared Client", and assigned to the present assignee, the specification of which is incorporated herein by reference.

Detailed Description Text (5):
The present invention provides a method and apparatus for distributing load in a multiple server computer environment by implementing a group manager process on each server, which periodically determines the server's capacity and load (i.e., utilization) with respect to multiple resources. The capacity and load information is broadcast to the other servers in the group, so that each server has a global view of every server's capacity and current load.

Detailed Description Text (6):
When a user attempts to access a DTU, the user inserts a unique identifier into the DTU. This identifier may be a smart card, password, biometric mechanism, or other mechanism which facilitates the identification of a user. Once the identifier is inserted, the DTU attempts to establish communications with the servers to start or resume one or more sessions. When a given DTU successfully starts or resumes a session, the group manager process of that server first determines whether one of the servers in the group already is hosting the session for that user. If that is the case, one embodiment redirects the DTU to that server and the load-balancing strategy is not employed. Otherwise, for each resource and server, the proper load balancing strategies are performed to identify which server is best able to handle

that particular session.

Detailed Description Text (47):
The invention implements multiple strategies that assign sessions to servers according to their capacity, current load, and the state of the multiple server environment (e.g., whether the arrival rate of new sessions is high or low). Referring now to FIG. 6, each server 600a and 600b runs a group manager process 601a and 601b when the software running on the server (e.g., the service) is started. The group manager processes 601a and 601b on each server periodically determines system capacity and utilization on each server 600a and 600b with respect to various resources, for instance, by querying the operating system.

Detailed Description Text (50):
Since group manager processes, 701a and 701b have periodically been communicating with each other regarding the sessions residing on each server, the group manager process 601a running on the server 701a, that the DTU 700 has attempted to establish communications with, will know if the session already resides on server 701b. If the session already resides on server 701b, in one embodiment server 701a redirects DTU 700 to server 701b via interconnect fabric 702 to switch 703 and across interconnect fabric 705 to server 701b where the session resides. In this case, the load-balancing strategy is not employed. Otherwise, for each resource and server, the relative desirability of assigning a new session to that server is computed.

Detailed Description Text (56):
A steady state exists when data is constant and the arrival rate of new sessions is low. One embodiment of the present invention implements the load balancing strategy in a steady state by determining the relative desirability of assigning a session to a server. The group manager obtains a ratio which represents the capacity of any given resource divided by the total current usage of the resource added to the total expected added usage by that resource.

Detailed Description Text (57):
In one embodiment, each group manager process residing on each server will independently compute the desirability and communicate the desirability to all other group manager processes. In another embodiment, each group manager process residing on each server will determine resource capacity and load. The group manager process will then pass resource utilization and load to all other group manager processes, which in turn will compute the desirabilities. In this way, each group manager process generates a global view of the system state. This strategy is illustrated by the following pseudo-code:

Detailed Description Text (58):
Referring now to FIG. 9, the group manager process 601 for a given server s determines resource capacity 901 with respect to server s. Next, the group manager process determines resource utilization 902 with respect to server s. Thereafter, the group manager process determines expected utilization by a session based on empirical measurements 903. In one embodiment, this measurement is obtained by the group manager process operating under the assumption that for a large group of sessions, average resource utilization by the sessions is relatively constant.

Detailed Description Text (59):
Once this data is obtained, the group manager process computes the desirability 904 of assigning a session to the given server s with respect to the tested resource r. Next, the group manager process determines if there are other resources to consider 905. Other resources may include, the number of microprocessors at a given server (since a server may have multiple processors), the number of sessions running on a given server relative to that server's carrying capacity (e.g., the maximum number of sessions that server can host), the states of sessions running on a server (e.g., active or inactive), and the unique expected requirements of certain users.

Detailed Description Text (60):
If there are other resources to consider, flow proceeds along transition 906 and the process is repeated. Otherwise, flow proceeds along transition 907 and the desirability of each resource with respect to server s is outputted 908. This strategy is repeated for each of the servers in the environment.

Detailed Description Text (62):
In one embodiment, a desirability value D is determined for each of three primary resources, which are, processing power (measured in CPU clock cycles), main memory (in megabytes of RAM) and network bandwidth (in megabits per second). Once the resource-specific desirability values are computed, the minimum of the three values is chosen for each server as the overall desirability. The motivation to pick the minimum value is that this represents the resource subject to the most contention on that server. This resource (the one that is most heavily loaded) represents the resource that is most likely to become saturated first, and is therefore, most important in deciding how to balance load.

Detailed Description Text (63):
The minimum of the resource specific desirability is obtained by implementing the following pseudo-code:

Detailed Description Text (64):
Once the minimum resource specific desirabilities are determined each server is ranked by level of desirability and the most desirable target can be selected.

Detailed Description Text (67):
Since the computational and memory resources allocated to the services requested by the DTUs are distributed across the group of servers, it is possible for resources to become unevenly allocated, thereby degrading performance on over-utilized servers while wasting resources on under-utilized servers. This is especially true in heterogeneous server configurations, where the carrying capacity of the servers (i.e., number and speed of processing units, amount of installed memory and available network bandwidth) is non-uniform. In this case, if a server with the lowest capacity responds first, all failed over DTUs may be redirected to this less powerful server, creating a seriously unbalanced load distribution and the risk of further server overload failures.

Detailed Description Text (70):
In this case, assignment to the least loaded server is not always the most effective method. In one embodiment, a pseudo-random strategy is used in unsteady states. FIG. 10a and 10b provides a flow control diagram of the pseudo-random strategy used in unsteady states. Initially, the group manager process 601 follows the steps provided in FIG. 9 to compute the desirability of assigning a session to a given server 908, by proceeding through steps 901, 902, 903, 904, 905, and 907 and transitioning at step 906 if other resources exist, or transitioning at step 907 if all resources have been tested.

Detailed Description Text (71):
Once all resources are determined, the strategy determines if other servers exist 1000. If so, flow transitions along 1001 and the process repeats by computing the desirability for other servers. If no other servers remain to be tested, flow transitions along path 1002 and the relative desirabilities are normalized to a predetermined range by the following steps. At step 1003 the sum of the desirabilities for each server is computed by adding each desirability. At step 1004, probabilistic weights are attached, by dividing the sum by the individual desirability. If there are other servers 1005, flow proceeds along transition 1006 and the process repeats for each server. When weights are determined for all available servers, each probabilistic weight is partitioned into a subrange z 1007. Thereafter, a random number is generated between 0 and z 1008. Whichever subrange

①

Best ⊙
even normalization

the random number falls into receives the session 1009.

Detailed Description Text (72):
Using this scheme, for instance, more desirable servers are weighted in a manner which makes it more likely to receive a fail over. Likewise, servers determined to be more heavily loaded receive a lower probability. Take for example, the case of two servers, server a with a desirability of 8 and server b with a desirability of 2. In this scenario, one embodiment of the invention will normalize these desirabilities to a range between 0 and 1. Therefore, server a will receive the range 0 to 0.8, while server b will receive the range between 0.8 and 1. A random number is generated between 0 and 1. The random number will determine assignment to the server, for instance if 0.6 is generated, server a gets the session. Thus, the pseudo-random strategy weighs the probability of any server s being selected by its computed desirability D[s]. Thus, in the provided example, if the session exists on server b, the group manager process on server a then sends a redirect message to the DTU, telling it to reconnect to the server b.

Detailed Description Text (74):
In one embodiment, the group manager can take into account the number of microprocessors at a given server when balancing load. With reference to FIG. 9, flow proceeds along steps 901-904, to determine the desirability by factoring in the number of microprocessors. In another embodiment, the number of sessions running on a given server relative to that server's carrying capacity (e.g., the maximum number of sessions that server can host) is a factor. Hence, in FIG. 9, flow transitions along steps 901-904. In another embodiment, the number of sessions running on a given server is a factor in computing desirability 904. An accurate snapshot of the number of sessions at any given time is known to the session manager running on the server. This embodiment may assume that over a large aggregate the type and amount of resources utilized by the sessions is similar and balances load based on this assumption.

Detailed Description Text (77):
In other embodiments, identifiers can be assigned to users which alert the system to the unique expected requirements of certain users. For instance, certain users are heavy users and routinely require CPU, memory, and bandwidth intensive resources while others may perform less intensive activities, such as checking e-mail. Therefore, the identity of the user can be used as an accurate method of determining the expected load on the system, for instance as a factor in steps 901-904 of FIG. 9.

Detailed Description Text (78):
In another embodiment the cost of redirection is considered. Redirection has a cost because to initiate a session on a server, the DTU must request authentication to the server, which can take up to several seconds. Once the server receives the authentication request, the group manager determines whether to accept the request, whether the session already exists on another server, in which case it will redirect the DTU to that server, or whether other servers have more available resources, which will cause the load balancing strategy to be employed.

Detailed Description Text (79):
If the result of the group manager's decision is to redirect the DTU to another server, the DTU must authenticate a second time on the selected target server. Since authentication takes a non-trivial amount of time, the cost of multiple authentications is considered in this embodiment. A variable factor is taken into account by the server in which the attempted authentication has occurred. For instance, the variable factor can be twenty percent. Therefore, the potential target server for redirection must have at least twenty percent more available resources than the attempted server. If the attempted and target servers are closer than the variable factor in their availability of a given resource, redirection does not occur and the cost associated with redirection is saved.

Detailed Description Text (80):
One embodiment of the present invention implements a hybrid strategy in employing the load balancing strategy. Referring to FIG. 11, the hybrid strategy combines the use of the steady state strategies, and the use of unsteady state strategies. Therefore, the information provided by the steady state load balancing strategy, such as processor, memory, or bandwidth utilization can be utilized when the group manager process determines that the server is in a steady state (i.e., the arrival rate of new sessions is low). The group manager process 601 determines if the server is in a steady or unsteady state 1100. If that the server is in an unsteady state (i.e., fail over causes the arrival rate of new sessions to instantly become large), flow proceeds along transition 1101 and the pseudo-random strategy 1102, and FIG. 10, is implemented to distribute failed over sessions more evenly among available resources 1105. If the server is in a steady state, (i.e., data is constant and the arrival rate of new sessions is low), flow proceeds along transition 1103 and the steady state strategy 1104, and FIG. 9, is implemented to distribute sessions more evenly among available resources 1105.

Detailed Description Paragraph Table (1):
Begin determine C[r,s], U[r,s], and A[r,s] where: C[r,s] = resource capacity the server has U[r,s] = resource utilization on the server at time of sampling A[r,s] = amount of the resource used on average by a session, based on empirical measurements for each resource r compute DR, the desirability of assigning a session to server s based on the resource-specific information where: DR[r,s] = C [r,s]/(U[r,s] + A[r,s]) end select DRmax as the maximum value of DR[r,s] over all servers, where: DRmax = max{s}(DR[r,s]) Normalize each DR[r,s] to values between 0 and 1 by dividing by the max value where: DR[r,s] = DR[r,s]/DRmax End.

Detailed Description Paragraph Table (2):
Begin for each server s select D[s] where: D[s] = the minimum value of DR[r,s] over all resources where: D[s] = min{r} (DR[r,s]) end End.

Other Reference Publication (1):
By Aman et al., "Adaptive Algorithms for Managing a Distributed Data Processing Workload", IBM Systems Journal, vol. 36, No. 2, 1997, pp. 242-283.

CLAIMS:

2. The method of claim 1, wherein determining said first and second desirabilities comprises determining an expected resource utilization.

3. The method of claim 1, wherein determining said first desirability comprises: determining a resource capacity; and evaluating a function of said resource capacity and said expected resource utilization.

4. The method of claim 3, wherein determining said second desirability comprises: determining a second resource capacity; and evaluating a second function of said second resource capacity and said expected resource utilization.

5. The method of claim 4, wherein said determining resource capacity, said determining expected resource utilization, and said evaluating steps are performed for each of a plurality of resources.

6. The method of claim 1, wherein said first server has a probability of selection weighted by said first desirability and said second server has a probability of selection weighted by said second desirability.

8. An apparatus comprising: a first server; a first station having a connection with said first server, wherein said first server is configured to determine an expected resource utilization; and a second server, wherein said first and said

second servers exchange information, and wherein said first server is configured to redirect said first station to said second server based on said information, wherein said first station is a client, said client providing an interface to a user and wherein each of said first and second servers can provide a plurality of computational services removed from said client to said user, and wherein said plurality of computational services comprise a computational power for said client and a state maintenance for said client.

10. The apparatus of claim 9, wherein said first server has a probability of selection weighted by said first desirability and said second server has a probability of selection weighted by said second desirability.

11. The apparatus of claim 8, wherein said first server is configured to: determine a resource capacity; and evaluate a function of said resource capacity and said expected resource utilization.

12. The apparatus of claim 8, wherein said first server is configured to: determine a resource capacity for each of a plurality of resources; determine a corresponding expected resource utilization for each of said resources; and evaluate a function for each of said resource capacities and said corresponding expected resource utilizations.

First Hit Fwd Refs

☐ Generate Collection Print

queue length

L6: Entry 3 of 4 File: USPT Dec 29, 1998

DOCUMENT-IDENTIFIER: US 5854754 A
TITLE: Scheduling computerized backup services

Brief Summary Text (5):
In a networked installation the client/server paradigm is often used to describe the roles of various components in the network. In a large installation, only a fraction of all clients can be backed up at the same time because the server and network cannot handle the load of all clients doing backup simultaneously. Traditionally, system administrators have manually tried different configuration alternatives to decide where to place the backup server machines in a proposed installation and which clients to connect to each backup server. This trial and error process is simplified if all the clients and all the workloads are of the same type, but becomes quite complex if clients are of different types and have different workloads.

Detailed Description Text (27):
workload total size;

Detailed Description Text (28):
workload number of files;

Detailed Description Text (29):
workload number of backup transactions generated, a characteristic of the average file size and the file size distribution; and

Detailed Description Text (46):
The system being modeled is composed of service centers (where requests queue for service), which are the networks, gateways, server CPU and server I/O devices, and of "waiting elements" (the clients or jobs in the system), which generate units of work. The service centers are characterized by service demand and the length of the queue of requests. Queues are commonly used in modeling to simulate processes where waiting time is involved. The installation can be pictured as a tree structure with each leaf generating a workload which is processed by the nodes (service centers). The processing by the nodes requires a finite amount of time. Graphically inputting a tree structure is one way of defining the structure to the CAP system. The root of the tree is the backup server. The workload generated during the backup process by each client will be processed by a certain number of service centers. For example, backing up data on a personal computer might require work by the personal computer, one or more networks/gateways through which the data must pass, the server CPU, and the server's I/O devices. The model maps these nodes into its service centers which give a measure of the times required to process the data from its home on a client to a backup I/O device.

Detailed Description Text (47):
In the following, the term "class" will be used to refer to a client type and a workload type. An example of a class would be all clients that are IBM PS/2 personal computers which have approximately 300 MB to back up, consisting of approximately 1000 files, and which are connected to a TokenRing network.

Detailed Description Text (48):

The term "workload" refers to all the work that the system needs to perform. In the backup system, this translates to all the files to be backed up by a set of clients. A job is a unit of work executed one at a time by each client. For example, units might be pages of data to be transmitted (for example, 4KB of data) or transactions to be finished. However, for the purpose of calculating the modeling parameters, the sizes of the complete client workload are used to keep the floating point arithmetic operations to a minimum. The final result depends only on the total size of the client workload as the individual job sizes cancel out in the calculations.

Detailed Description Text (52):
After all these calculations are done, we have the values of D.sub.ij, the service demand of class i at service center j 21. Each queue length is given an initial estimate, the same for all queues, and it is the total number of clients in the system divided by the total number of service centers 22. The next thing to do is to estimate the throughput of the whole system, which is the calculation of the queue length q.sub.ij of each class i at each service center j, because it determines the time spent waiting at each service center 23. Once we know these times, we calculate the total time spent by a client job in the complete system, by adding all the times spent waiting and executing at each service center 24. Next, the value of the queue length of each class at each service center is calculated by an iterative process 25. Each iteration yields a new value of the queue length q.sub.ij. This value is compared with the previous value of the queue length, and the loop is repeated until the values converge by differing by less than an acceptable value.

Detailed Description Text (53):
Inside this iteration loop there is an additional loop over each class: for each class i we calculate the time Q.sub.ij spent at each service center j 23. This time is calculated as the service demand D.sub.ij times the number of jobs in the queue (q.sub.j +1-q.sub.ij /N). The total queue length at the service center is q.sub.j, and the queue length of the class i is q.sub.ij. The correction factor q.sub.ij /N accounts for the fact that the client does not see itself waiting in the queue, and the 1 accounts for the client itself. Then, we calculate the response time R.sub.i, as the sum of all times Q.sub.ij spent at all service centers, plus the "waiting time" Z.sub.i at the client 24. Then, we calculate the class throughput X.sub.i using Little's law, as the number of clients of this class N.sub.i divided by the response time R.sub.i 24. Then, we recalculate q.sub.ij as the product of the class throughput X.sub.i and the time spent at device j, Q.sub.ij 25. We add this new value to the total queue length q.sub.j, and continue to iterate 25. For each service center, the error is calculated in a standard manner as the absolute value of the difference between the new and old values of the queue length, divided by the old value. The exit criterion is that each error has to be less than a preset value 26. For robustness, there can also be an exit criterion after a preset number of iterations through the loop, to prevent an infinite loop.

Detailed Description Text (68):
There is also the possibility that the time window to do the backup is smaller than the shortest backup time of any group. In this case the problem has no solution as stated. The system administrator has to modify the base configuration and then retry using CAP. Possible changes that the administrator can use include decreasing the workload on each machine, decreasing the number of clients per network, upgrading the clients to be faster models, and upgrading the networks or the servers.

Detailed Description Text (89):
Consider the enterprise structure shown in FIG. 1, which is similar to the big enterprise described above. In the following, specific current hardware and software will be assigned to the generic components for purposes of illustration; but any computers capable of acting as servers could be substituted by altering the

associated characteristics used for the calculations. Assume that the server is an RS/6000 workstation with four disk drives attached. "Network 1" is a collection of four interconnected Token Ring (TR) networks and "Network 2" is a collection of four interconnected Ethernet (ET) networks. There are four TR networks and four ET networks. Each TR network has two classes of clients attached: the first class, "Client 1" in FIG. 1, is 20 RS/6000 workstations (AIX-TR) per network, or a total of 80. The second class, "Client 2" in FIG. 1, is 80 PS/2 personal computers (OS/2-TR) per network, or a total of 320. Each ET network has also two classes of clients attached: the first class, "Client 3" in FIG. 1, is 10 RS/6000 workstations (AIX-ET) per network, or a total of 40. The second class, "Client 4" in FIG. 1, is 40 PS/2 personal computers (OS/2-ET) per network, or a total of 160. The characteristics of each class, including its workload (WL), are shown in Table 1. The suggested utilization at the server and network is 80% and the time window available for backup is six hours.

Detailed Description Text (90):
The total number of network types in the enterprise is two, and the total number of networks is 2.times.4=8. The total number of client types in the enterprise is four, and the total number of clients is 4.times.(20+80)+4.times.(10+40)=600. The total workload size per enterprise is 248 GB.

Detailed Description Text (95):
Next, we calculate the number of networks that can participate in the first group, for the network type at the top of the list--the TR network: we divide the service demand at the network (Table 3, column 2) by the service demand at the server (Table 3, column 3), normalized by the respective suggested utilizations at the network and server (both 80% in our case):

CLAIMS:

1. A method for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the method comprising the steps of:

(a) building a model of the computer installation which calculates elapsed time and utilization of computer installation resources for the selected service for a selected subset of clients for the selected service for a selected subset of clients, the model using definitions of client types, network types and interconnection of clients and networks in the computer installation;

(b) repeatedly invoking the model with a minimum number of clients and then a maximum number of clients of a single type to find elapsed times and utilization of computer installation resources, then invoking the model with varying numbers of clients between the minimum and maximum numbers of clients until finding a number of clients for the subset for which the utilization or elapsed time criteria are met and adjusting the subsets to contain clients which can be serviced sequentially without exceeding an elapsed time criterion or a utilization criterion; and

(c) generating a schedule by arranging the subsets into a sequence.

3. An apparatus for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, comprising:

(a) a Modeler which calculates utilization of computer installation resources and elapsed time for the selected service for a selected subset of clients, the calculations using definitions of client types, network types and interconnection of clients and networks in the computer installation, using queues to model elements in the installation which work to provide the services and using Little's

Law to calculate throughput; and

(b) a Scheduler which repeatedly invokes the Modeler with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

6. An apparatus for scheduling services in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, comprising:

(a) first means for modeling utilization of the computer installation resources for services for a selected subset of clients, which calculates utilization of installation resources using definitions of client types, network types and interconnection of clients and networks in the computer installation;

(b) second means for modeling which calculates elapsed time for the selected service for a selected subset of clients using Little's Law to calculate throughput;

(c) means for scheduling which repeatedly invokes the first and second means for modeling with subsets of clients to find utilizations and elapsed times for the subsets and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

10. A computer readable storage device containing program code for scheduling a selected service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the program code comprising:

(a) program code for modeling utilization of the computer installation resources and elapsed time for the selected service for a selected subset of clients uses Little's Law to calculate throughout, the modeling using definitions of client types, network types and interconnection of clients and networks in the computer installation; and

(b) program code for scheduling which repeatedly invokes the program code for modeling with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

L6: Entry 3 of 4 File: USPT Dec 29, 1998

DOCUMENT-IDENTIFIER: US 5854754 A
TITLE: Scheduling computerized backup services

Brief Summary Text (5):
In a networked installation the client/server paradigm is often used to describe the roles of various components in the network. In a large installation, only a fraction of all clients can be backed up at the same time because the server and network cannot handle the load of all clients doing backup simultaneously. Traditionally, system administrators have manually tried different configuration alternatives to decide where to place the backup server machines in a proposed installation and which clients to connect to each backup server. This trial and error process is simplified if all the clients and all the workloads are of the same type, but becomes quite complex if clients are of different types and have different workloads.

Detailed Description Text (27):
workload total size;

Detailed Description Text (28):
workload number of files;

Detailed Description Text (29):
workload number of backup transactions generated, a characteristic of the average file size and the file size distribution; and

Detailed Description Text (46):
The system being modeled is composed of service centers (where requests queue for service), which are the networks, gateways, server CPU and server I/O devices, and of "waiting elements" (the clients or jobs in the system), which generate units of work. The service centers are characterized by service demand and the length of the queue of requests. Queues are commonly used in modeling to simulate processes where waiting time is involved. The installation can be pictured as a tree structure with each leaf generating a workload which is processed by the nodes (service centers). The processing by the nodes requires a finite amount of time. Graphically inputting a tree structure is one way of defining the structure to the CAP system. The root of the tree is the backup server. The workload generated during the backup process by each client will be processed by a certain number of service centers. For example, backing up data on a personal computer might require work by the personal computer, one or more networks/gateways through which the data must pass, the server CPU, and the server's I/O devices. The model maps these nodes into its service centers which give a measure of the times required to process the data from its home on a client to a backup I/O de ice. v

Detailed Description Text (47):
In the following, the term "class" will be used to refer to a client type and a workload type. An example of a class would be all clients that are IBM PS/2 personal computers which have approximately 300 MB to back up, consisting of approximately 1000 files, and which are connected to a TokenRing network.

Detailed Description Text (48):

The term "workload" refers to all the work that the system needs to perform. In the backup system, this translates to all the files to be backed up by a set of clients. A job is a unit of work executed one at a time by each client. For example, units might be pages of data to be transmitted (for example, 4KB of data) or transactions to be finished. However, for the purpose of calculating the modeling parameters, the sizes of the complete client workload are used to keep the floating point arithmetic operations to a minimum. The final result depends only on the total size of the client workload as the individual job sizes cancel out in the calculations.

Detailed Description Text (52):
After all these calculations are done, we have the values of D.sub.ij, the service demand of class i at service center j 21. Each queue length is given an initial estimate, the same for all queues, and it is the total number of clients in the system divided by the total number of service centers 22. The next thing to do is to estimate the throughput of the whole system, which is the calculation of the queue length q.sub.ij of each class i at each service center j, because it determines the time spent waiting at each service center 23. Once we know these times, we calculate the total time spent by a client job in the complete system, by adding all the times spent waiting and executing at each service center 24. Next, the value of the queue length of each class at each service center is calculated by an iterative process 25. Each iteration yields a new value of the queue length q.sub.ij. This value is compared with the previous value of the queue length, and the loop is repeated until the values converge by differing by less than an acceptable value.

Detailed Description Text (53):
Inside this iteration loop there is an additional loop over each class: for each class i we calculate the time Q.sub.ij spent at each service center j 23. This time is calculated as the service demand D.sub.ij times the number of jobs in the queue (q.sub.j +1-q.sub.ij /N). The total queue length at the service center is q.sub.j, and the queue length of the class i is q.sub.ij. The correction factor q.sub.ij /N accounts for the fact that the client does not see itself waiting in the queue, and the 1 accounts for the client itself. Then, we calculate the response time R.sub.i, as the sum of all times Q.sub.ij spent at all service centers, plus the "waiting time" Z.sub.i at the client 24. Then, we calculate the class throughput X.sub.i using Little's law, as the number of clients of this class N.sub.i divided by the response time R.sub.i 24. Then, we recalculate q.sub.ij as the product of the class throughput X.sub.i and the time spent at device j, Q.sub.ij 25. We add this new value to the total queue length q.sub.j, and continue to iterate 25. For each service center, the error is calculated in a standard manner as the absolute value of the difference between the new and old values of the queue length, divided by the old value. The exit criterion is that each error has to be less than a preset value 26. For robustness, there can also be an exit criterion after a preset number of iterations through the loop, to prevent an infinite loop.

Detailed Description Text (68):
There is also the possibility that the time window to do the backup is smaller than the shortest backup time of any group. In this case the problem has no solution as stated. The system administrator has to modify the base configuration and then retry using CAP. Possible changes that the administrator can use include decreasing the workload on each machine, decreasing the number of clients per netw rk, upgrading the clients to be faster models, and upgrading the networks or the servers.

Detailed Description Text (89):
Consider the enterprise structure shown in FIG. 1, which is similar to the big enterprise described above. In the following, specific current hardware and software will be assigned to the generic components for purposes of illustration; but any computers capable of acting as servers could be substituted by altering the

associated characteristics used for the calculations. Assume that the server is an RS/6000 workstation with four disk drives attached. "Network 1" is a collection of four interconnected Token Ring (TR) networks and "Network 2" is a collection of four interconnected Ethernet (ET) networks. There are four TR networks and four ET networks. Each TR network has two classes of clients attached: the first class, "Client 1" in FIG. 1, is 20 RS/6000 workstations (AIX-TR) per network, or a total of 80. The second class, "Client 2" in FIG. 1, is 80 PS/2 personal computers (OS/2-TR) per network, or a total of 320. Each ET network has also two classes of clients attached: the first class, "Client 3" in FIG. 1, is 10 RS/6000 workstations (AIX-ET) per network, or a total of 40. The second class, "Client 4" in FIG. 1, is 40 PS/2 personal computers (OS/2-ET) per network, or a total of 160. The characteristics of each class, including its workload (WL), are shown in Table 1. The suggested utilization at the server and network is 80% and the time window available for backup is six hours.

Detailed Description Text (90):
The total number of network types in the enterprise is two, and the total number of networks is 2.times.4=8. The total number of client types in the enterprise is four, and the total number of clients is 4.times.(20+80)+4.times.(10+40)=600. The total workload size per enterprise is 248 GB.

Detailed Description Text (95):
Next, we calculate the number of networks that can participate in the first group, for the network type at the top of the list--the TR network: we divide the service demand at the network (Table 3, column 2) by the service demand at the server (Table 3, column 3), normalized by the respective suggested utilizations at the network and server (both 80% in our case):

CLAIMS:

1. A method for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the method comprising the steps of:

(a) building a model of the computer installation which calculates elapsed time and utilization of computer installation resources for the selected service for a selected subset of clients for the selected service for a selected subset of clients, the model using definitions of client types, network types and interconnection f olients and networks in the computer installation;

(b) repeatedly invoking the model with a minimum number of clients and then a maximum number of clients of a single type to find elapsed times and utilization of computer installation resources, then invoking the model with varying numbers of clients between the minimum and maximum numbers of clients until finding a number of clients for the subset for which the utilization or elapsed time criteria are met and adjusting the subsets to contain clients which can be serviced sequentially without exceeding an elapsed time criterion or a utilization criterion; and

(c) generating a schedule by arranging the subsets into a sequence.

3. An apparatus for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of i terconnected networks and at least one server, comprising:

(a) a Modeler which calculates utilization of computer installation resources and elapsed time for the selected service for a selected subset of clients, the calculations using definitions of client types, network types and interconnection of clients and networks in the computer installation, using queues to model elements in the installation which work to provide the services and using Little's

Law to calculate throughput; and

(b) a Scheduler which repeatedly invokes the Modeler with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

6. An apparatus for scheduling services in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, comprising:

(a) first means for modeling utilization of the computer installation resources for services for a selected subset of clients, which calculates utilization of installation resources using definitions of client types, network types and interconnection of clients and networks in the computer installation;

(b) second means for modeling which calculates elapsed time for the selected service for a selected subset of clients using Little's Law to calculate throughput;

(c) means for scheduling which repeatedly invokes the first and second means for modeling with subsets of clients to find utilizations and elapsed times for the subsets and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

10. A computer readable storage device containing program code for scheduling a selected service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the program code comprising:

(a) program code for modeling utilization of the computer installation resources and elapsed time for the selected service for a selected subset of clients uses Little's Law to calculate throughout, the modeling using definitions of client types, network types and interconnection of clients and networks in the computer installation; and

(b) program code for scheduling which repeatedly invokes the program code for modeling with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

L6: Entry 3 of 4 File: USPT Dec 29, 1998

DOCUMENT-IDENTIFIER: US 5854754 A
TITLE: Scheduling computerized backup services

Brief Summary Text (5):
In a networked installation the client/server paradigm is often used to describe the roles of various components in the network. In a large installation, only a fraction of all clients can be backed up at the same time because the server and network cannot handle the load of all clients doing backup simultaneously. Traditionally, system administrators have manually tried different configuration alternatives to decide where to place the backup server machines in a proposed installation and which clients to connect to each backup server. This trial and error process is simplified if all the clients and all the workloads are of the same type, but becomes quite complex if clients are of different types and have different workloads.

Detailed Description Text (27):
workload total size;

Detailed Description Text (28):
workload number of files;

Detailed Description Text (29):
workload number of backup transactions generated, a characteristic of the average file size and the file size distribution; and

Detailed Description Text (46):
The system being modeled is composed of service centers (where requests queue for service), which are the networks, gateways, server CPU and server I/O devices, and of "waiting elements" (the clients or jobs in the system), which generate units of work. The service centers are characterized by service demand and the length of the queue of requests. Queues are commonly used in modeling to simulate processes where waiting time is involved. The installation can be pictured as a tree structur with each leaf generating a workload which is processed by the nodes (service centers). The processing by the nodes requires a finite amount of time. Graphically inputting a tree structure is one way of defining the structure to the CAP system. The root of the tree is the backup server. The workload generated during the backup process by each client will be processed by a certain number of service centers. For example, backing up data on a personal computer might require work by the personal computer, one or more networks/gateways through which the data must pass, the server CPU, and the server's I/O devices. The model maps these nodes into its service centers which give a measure of the times required to process the data from its home on a client to a backup I/O de ice.

Detailed Description Text (47):
In the following, the term "class" will be used to refer to a client type and a workload type. An example of a class would be all clients that are IBM PS/2 personal computers which have approximately 300 MB to back up, consisting of approximately 1000 files, and which are connected to a TokenRing network.

Detailed Description Text (48):

The term "workload" refers to all the work that the system needs to perform. In the backup system, this translates to all the files to be backed up by a set of clients. A job is a unit of work executed one at a time by each client. For example, units might be pages of data to be transmitted (for example, 4KB of data) or transactions to be finished. However, for the purpose of calculating the modeling parameters, the sizes of the complete client workload are used to keep the floating point arithmetic operations to a minimum. The final result depends only on the total size of the client workload as the individual job sizes cancel out in the calculations.

Detailed Description Text (52):
After all these calculations are done, we have the values of D.sub.ij, the service demand of class i at service center j 21. Each queue length is given an initial estimate, the same for all queues, and it is the total number of clients in the system divided by the total number of service centers 22. The next thing to do is to estimate the throughput of the whole system, which is the calculation of the queue length q.sub.ij of each class i at each service center j, because it determines the time spent waiting at each service center 23. Once we know these times, we calculate the total time spent by a client job in the complete system, by adding all the times spent waiting and executing at each service center 24. Next, the value of the queue length of each class at each service center is calculated by an iterative process 25. Each iteration yields a new value of the queue length q.sub.ij. This value is compared with the previous value of the queue length, and the loop is repeated until the values converge by differing by less than an acceptable value.

Detailed Description Text (53):
Inside this iteration loop there is an additional loop over each class: for each class i we calculate the time Q.sub.ij spent at each service center j 23. This time is calculated as the service demand D.sub.ij times the number of jobs in the queue (q.sub.j +1-q.sub.ij /N). The total queue length at the service center is q.sub.j, and the queue length of the class i is q.sub.ij. The correction factor q.sub.ij /N accounts for the fact that the client does not see itself waiting in the queue, and the 1 accounts for the client itself. Then, we calculate the response time R.sub.i, as the sum of all times Q.sub.ij spent at all service centers, plus the "waiting time" Z.sub.i at the client 24. Then, we calculate the class throughput X.sub.i using Little's law, as the number of clients of this class N.sub.i divided by the response time R.sub.i 24. Then, we recalculate q.sub.ij as the product of the class throughput X.sub.i and the time spent at device j, Q.sub.ij 25. We add this new value to the total queue length q.sub.j, and continue to iterate 25. For each service center, the error is calculated in a standard manner as the absolute value of the difference between the new and old values of the queue length, divided by the old value. The exit criterion is that each error has to be less than a preset value 26. For robustness, there can also be an exit criterion after a preset number of iterations through the loop, to prevent an infinite loop.

Detailed Description Text (68):
There is also the possibility that the time window to do the backup is smaller than the shortest backup time of any group. In this case the problem has no solution as stated. The system administrator has to modify the base configuration and then retry using CAP. Possible changes that the administrator can use include decreasing the workload on each machine, decreasing the number of clients per network, upgrading the clients to be faster models, and upgrading the networks or the servers.

Detailed Description Text (89):
Consider the enterprise structure shown in FIG. 1, which is similar to the big enterprise described above. In the following, specific current hardware and software will be assigned to the generic components for purposes of illustration; but any computers capable of acting as servers could be substituted by altering the

associated characteristics used for the calculations. Assume that the server is an RS/6000 workstation with four disk drives attached. "Network 1" is a collection of four interconnected Token Ring (TR) networks and "Network 2" is a collection of four interconnected Ethernet (ET) networks. There are four TR networks and four ET networks. Each TR network has two classes of clients attached: the first class, "Client 1" in FIG. 1, is 20 RS/6000 workstations (AIX-TR) per network, or a total of 80. The second class, "Client 2" in FIG. 1, is 80 PS/2 personal computers (OS/2-TR) per network, or a total of 320. Each ET network has also two classes of clients attached: the first class, "Client 3" in FIG. 1, is 10 RS/6000 workstations (AIX-ET) per network, or a total of 40. The second class, "Client 4" in FIG. 1, is 40 PS/2 persona computers (OS/2-ET) per network, or a total of 160. The characteristics of each class, including its workload (WL), are shown in Table 1. The suggested utilization at the server and network is 80% and the time window available for backup is six hours.

Detailed Description Text (90):
The total number of network types in the enterprise is two, and the total number of networks is 2.times.4=8. The total number of client types in the enterprise is four, and the total number of clients is 4.times.(20+80)+4.times.(10+40)=600. The total workload size per enterprise is 248 GB.

Detailed Description Text (95):
Next, we calculate the number of networks that can participate in the first group, for the network type at the top of the list--the TR network: we divide the service demand at the network (Table 3, column 2) by the service demand at the server (Table 3, column 3), normalized by the respective suggested utilizations at the network and server (both 80% in our case):

CLAIMS:

1. A method for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the method comprising the steps of:

(a) building a model of the computer installation which calculates elapsed time and utilization of computer installation resources for the selected service for a selected subset of clients for the selected service for a selected subset of clients, the model using definitions of client types, network types and interconnection of clients and networks in the computer installation;

(b) repeatedly invoking the model with a minimum number of clients and then a maximum number of clients of a single type to find elapsed times and utilization of computer installation resources, then invoking the model with varying numbers of clients between the minimum and maximum numbers of clients until finding a number of clients for the subset for which the utilization or elapsed time criteria are met and adjusting the subsets to contain clients which can be serviced sequentially without exceeding an elapsed time criterion or a utilization criterion; and

(c) generating a schedule by arranging the subsets into a sequence.

3. An apparatus for scheduling a backup service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, comprising:

(a) a Modeler which calculates utilization of computer installation resources and elapsed time for the selected service for a selected subset of clients, the calculations using definitions of client types, network types and interconnection of clients and networks in the computer installation, using queues to model elements in the installation which work to provide the services and using Little's

Law to calculate throughput; and

(b) a Scheduler which repeatedly invokes the Modeler with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

6. An apparatus for scheduling services in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, comprising:

(a) first means for modeling utilization of the computer installation resources for services for a selected subset of clients, which calculates utilization of installation resources using definitions of client types, network types and interconnection of clients and networks in the computer installation;

(b) second means for modeling which calculates elapsed time for the selected service for a selected subset of clients using Little's Law to calculate throughput;

(c) means for scheduling which repeatedly invokes the first and second means for modeling with subsets of clients to find utilizations and elapsed times for the subsets and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

10. A computer readable storage device containing program code for scheduling a selected service in a computer installation having a plurality of clients consisting of more than one client type, a plurality of interconnected networks and at least one server, the program code comprising:

(a) program code for modeling utilization of the computer installation resources and elapsed time for the selected service for a selected subset of clients uses Little's Law to calculate throughout, the modeling using definitions of client types, network types and interconnection of clients and networks in the computer installation; and

(b) program code for scheduling which repeatedly invokes the program code for modeling with subsets of clients to find utilizations and elapsed times and adjusts the subsets to generate a schedule which is a list of subsets of clients which can be serviced sequentially without exceeding a utilization criterion or an elapsed time criterion.

First Hit Fwd Refs

2 9

Workload Manage ment

based on metric

polling

L60

☐ Generate Collection Print

L38: Entry 1 of 5 File: USPT Jun 3, 2003

DOCUMENT-IDENTIFIER: US 6574587 B2
TITLE: System and method for extracting and forecasting computing resource data such as CPU consumption using autoregressive methodology

Abstract Text (1):
A system and method for extracting and forecasting computing resource data such as workload consumption of mainframe computing resources using an autoregressive model. The system and method forecast mainframe central processing unit (CPU) consumption with ninety-five percent accuracy using historical performance data. The system and method also provide an upper ninety-five percent confidence level and a lower ninety-five percent confidence level. The system and method retrieve performance records from a computer platform in one second intervals, statistically collapses the one second performance data into fifteen minute performance data, statistically collapses the fifteen minute performance data into one week performance data, and generates a time series equivalent to collecting performance data at one week intervals. The system and method ensure that the resulting time series is statistically stationary, and applies an autoregressive construct to the time series to generate forecast of future CPU utilization, as well as to generate reports and graphs comparing actual vs. forecast CPU utilization. Because the system and method rely on electronically generated empirical historical computer performance data as an input, they provide a turnkey solution to CPU consumption forecasting that can be implemented easily by any system network manager.

Brief Summary Text (2):
The present invention relates to a computer platform, and in particular, to a system and method to forecast the performance of computing resources.

Brief Summary Text (4):
The computing resources of a large business represent a significant financial investment. When the business grows, resource managers must ensure that new resources are added as processing requirements increase. The fact that the growth and evolution of a computing platform is often rapid and irregular complicates management efforts. This is especially true for computing platforms common to banking institutions and telecommunications companies, for example, whose computing platforms typically include hundreds of geographically distributed computers.

Brief Summary Text (5):
To effectively manage the vast resources of a computing platform and to justify any requests for acquisition of new resources, managers need accurate forecasts of computing platform resource performance. However, conventional forecasting tools may not be adequate for use on computing platforms. For example, conventional sales performance forecasting tools, which use linear regression and multivariable regression to analyze data, commonly factor in such causal variables as the effect of holiday demand, advertising campaigns, price changes, etc. Similarly, pollution forecasting tools typically consider the causal effect of variations in traffic patterns. As such, using these tools to forecast computing platform resources may be problematical because causal parameters generally are difficult to establish and are unreliable.

Brief Summary Text (7):

The limitations of established forecasting tools are particularly troublesome when forecasting resources in computing platforms that are expanding or are already re-engineered. These computing platforms need a forecasting system and method that deal appropriately with new data as well as unneeded data. Moreover, these computing platforms need a forecasting system and method that augment causal-based forecasting tools to provide accurate and reliable forecasts.

Brief Summary Text (9):
Presented herein is a system and method to forecast computing platform resource performance that overcomes the limitations associated with conventional forecasting tools. An embodiment applies an autoregressive model to electronically generated empirical data to produce accurate and reliable computing platform resource performance forecasts. An embodiment of the present invention also statistically collapses large amounts of data, eliminates unneeded data, and recursively processes new data. The forecasts are compared to actual performance data, which may be graphically displayed or printed. A specific type of data is not important for the present invention, and those skilled in the art will understand that a wide variety of data may be used in the present invention. For example, the present invention contemplates any data that may be collected and verified over time. These data include, for example, Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc.

Brief Summary Text (11):
The computing platform includes at least one resource whose performance is forecast. In one embodiment, the computing platform resource may be a central processing unit (CPU). In another embodiment, the computing platform resource may be a memory storage unit. In other embodiments, the computing platform resource may be a printer, a disk, or a disk drive unit. A specific computing platform resource is not important for the present invention, and those skilled in the art will understand that a number of resources may be used in the present invention.

Brief Summary Text (12):
Each resource includes at least one aspect. The aspect may be a performance metric. The performance metric may be resource utilization. "Utilization" is defined generally herein as the percentage that a particular computing platform resource is kept busy. Utilization is often termed "consumption."

Brief Summary Text (13):
In another embodiment, the performance metric may be resource efficiency or resource redundancy. "Efficiency" is defined generally herein as the measure of the useful portion of the total work performed by the resource. "Redundancy" is defined generally herein as the measure of the increase in the workload of a particular resource. Of course, those skilled in the art will appreciate that a particular performance metric is not required by the present invention. Instead, a number of performance metrics may be used.

Brief Summary Text (14):
In one embodiment, the computing platform includes a resource manager. The resource manager collects performance data from its associated resource. The performance data is associated with a performance metric. In one embodiment, the resource manager collects performance data representing a CPU utilization performance metric.

Brief Summary Text (15):
The resource manager collects the performance data in regular intervals. In one embodiment, regular intervals include one-second intervals, for example. That is, in this embodiment, the resource manager collects performance data from its associated computer(s) every second. The interval size in which performance data is

collected may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (17):
A first statistical collapser generates a first time series representing a performance metric as though its associated performance data had been collected at a first interval. The first time series includes a first set of time points. In one embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected in fifteen minute intervals. Accordingly, the time series includes four time points for each hour. In another embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected hourly. Accordingly, the time series includes one time point for each hour. It will be understood by persons skilled in the relevant art that the present invention encompasses statistical collapsers that generate time series representing performance metrics as though their associated performance data had been collected at any of a variety of suitable intervals. The interval size and corresponding number of time points generated by the first statistical collapser may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (20):
The computing platform also includes a second statistical collapser. The second statistical collapser statistically collapses the first time series, producing a second time series. The second time series includes a second set of time points. In one embodiment, the second statistical collapser statistically collapses the fifteen minute time series into a one-week time series. That is, the second statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected weekly. Accordingly, the time series includes approximately four time points for each month. In another embodiment, the second statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected daily. The corresponding time series includes approximately thirty time points for each month. It will be understood by persons skilled in the relevant art that the second statistical collapser may generate time series representing a performance metric as though its performance data had been collected at any of a variety of suitable intervals. As described above with reference to the first statistical collapser, the interval size and corresponding number of time points generated by the second statistical collapser may be determined by the particular use for the performance metric, the particular resource, the particular computing platform, etc.

Brief Summary Text (23):
One feature of the present invention is an autoregressive modeling tool, which is applied to the converted time series to forecast a particular aspect of the computing platform. The autoregressive modeling tool is chosen by calculating autocorrelation, inverse autocorrelation, and partial autocorrelation functions, and by comparing these functions to theoretical correlation functions of several autoregressive constructs. In particular, one embodiment applies a first order mixed autoregressive construct, such as an autoregressive moving average (ARMA) construct, to the differenced time series. Another embodiment applies an autoregressive integrated moving average (ARIMA) construct to the differenced time series. In the embodiment where the performance metric is resource utilization and the resource is a CPU, the resulting autoregressive modeling tool reliably forecasts CPU consumption with a ninety-five percent accuracy, provides an upper ninety-five percent confidence level, and provides a lower ninety-five percent confidence level. Conventional systems and methods that rely on linear regression or multivariable regression techniques may carry a lower confidence level.

Brief Summary Text (24):

Another feature of the present invention is that it uses empirical data as inputs to the autoregressive modeling tool. Using empirical data rather than causal variables provides more accurate forecasts. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit, the empirical data is actual historical performance data, including logical CPU utilization information as well as physical CPU utilization information. Moreover, the system and method generate recursive forecasts whereby actual future performance data is fed back into the autoregressive modeling tool to calibrate the autoregressive modeling tool.

Brief Summary Text (26):
In one embodiment, the results processor may be a graphical display unit, such as a computer display screen. In another embodiment, the results processor may be a textual display unit, such as a printer. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit, the results processor produces reports and graphical representations of comparisons of actual CPU utilization with CPU utilization forecasts.

Detailed Description Text (2):
A computer platform, and in particular, a system and method for forecasting computer platform resource performance is described herein. In the following description, numerous specific details, such as specific statistical symbols and relationships, specific methods of analyzing and processing computer performance data, etc., are set forth in order to provide a full understanding of the present invention. One skilled in the relevant art, however, will readily recognize that the present invention can be practiced without one or more of the specific details, or with other methods, etc. In other instances, well-known structures or operations are not shown in detail in order to avoid obscuring the present invention.

Detailed Description Text (3):
For illustrative purposes, embodiments of the present invention are sometimes described with respect to a system and method for forecasting computer platform resource performance. It should be understood that the present invention is not limited to these embodiments. Instead, the present invention contemplates any data that may be collected and verified over time. These data may include, for example, Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc.

Detailed Description Text (9):
The computing platform 100 includes at least one resource. In one embodiment, the computing platform resource may be a central processing unit (CPU). In another embodiment, the computing platform resource may be a memory storage unit. In other embodiments, the computing platform resource may be a printer, a disk, or a disk drive unit. While a specific computing platform resource is not important for the present invention, those skilled in the art will understand that any number of resources can be used in the present invention.

Detailed Description Text (10):
Each resource includes at least one aspect. The aspect may be a performance metric. In one embodiment the performance metric may be resource utilization. Utilization is the measure of the percentage that a particular computing platform resource is kept busy, and is sometimes termed consumption. In another embodiment, the performance metric may be resource efficiency, which is defined as the measure of the useful portion of the total work performed by the resource. In another embodiment, the performance metric may be resource redundancy, which is defined as the measure of the increase in the workload of a particular resource. Of course, those skilled in the art will appreciate that a particular performance metric is not required by the present invention. Instead, the present invention supports any

of a number of performance metrics.

Detailed Description Text (11):
FIG. 2 is a more detailed block diagram of the computing platform 100 according to one embodiment. As illustrated, each computer 106 includes a resource manager 202. Each resource manager 202 collects performance data from its associated resource. The performance data is associated with a performance metric. According to one embodiment, the resource manager 202 is a resource management facility (RMF) available with the multiple virtual storage (MVS) operating system that is running on the IBM mainframe computer as noted above or an equivalent mainframe computer available from Amdahl and Hitachi Data Systems. According to this embodiment, the resource manager 202 extracts historical performance data from a processor resource/systems manager (PR/SM) (not shown) of the computer 106. This historical computer performance data represents the CPU utilization and is equivalent to performance metering data obtained by real-time monitors. Thus, the CPU utilization information collected by the resource manager 202 are CPU utilization records that contain CPU activity measurements.

Detailed Description Text (12):
The resource manager 202 collects the performance data from the computer 106 at regular intervals. According to an exemplary embodiment, the regular intervals are one-second intervals. That is, according to the exemplary embodiment, the resource manager collects CPU workload performance data every second from computer 106. In this way, the resource manager 202 provides the percent busy for each computer 106 each second in time. The interval size in which performance data is collected may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc.

Detailed Description Text (13):
Because the computers 106 typically are maintained by large entities, the amount of data collected usually is quite large. Consequently, the data must be reduced to a manageable level. Statistically collapsing the one-second records generated by the resource manager 202 serves this purpose. The computing platform 100 thus also includes a plurality of statistical collapsers that statistically collapse the performance data into time series representing a performance metric. A "time series" is defined herein generally as any ordered sequence of observations. Each observation represents a given point in time and is thus termed a "time point." A statistical collapser averages a series of time points and generates a time series representing a performance metric as though its associated performance data had been collected at a particular interval. The resulting time series contains a set of time points commensurate with the representative collection interval.

Detailed Description Text (14):
According to one embodiment, the computing platform 100 includes a statistical collapser 204 that statistically collapses the performance data collected by the resource manager 202 into a time series. The statistical collapser 204 generates a time series representing performance data as though it had been collected at fifteen-minute intervals. Accordingly, the time series would include four time points for each hour. In another embodiment, the first statistical collapser generates a time series representing a performance metric as though its associated performance data had been collected hourly. Accordingly, the time series would include one time point for each hour.

Detailed Description Text (15):
Thus, the statistical collapser 204 statistically collapses the CPU utilization records generated every second by the resource manager 202 into CPU utilization records representing fifteen-minute intervals. Nine hundred original CPU utilization records ([60 seconds/minute].times.[15 minutes]=900) are averaged to produce one collapsed time point. The statistical collapser 204 calculates the mean for all metering records collected by the resource manager 202, as described in

greater detail below. The statistical collapser 204 then determines the median for each mean at fifteen-minute intervals. The time series generated by the statistical collapser 204 thus consists of four data points (or time points) per hour representing the mean CPU utilization percentage. It will be understood by persons skilled in the relevant art that the present invention encompasses statistical collapsers that generate time series representing performance metrics as though its associated performance data had been collected at any of a variety of suitable intervals. The interval size and corresponding number of time points generated by the statistical collapser may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc.

Detailed Description Text (16):
A stochastic process, such as the time series representing the performance metric as though its performance data had been collected at fifteen-minute intervals, may be represented by Z(.omega.,t). As used herein, a stochastic process is generally a family of time indexed random variables, Z(.omega.,t), where .omega. belongs to a sample space and t belongs to a time index set. That is, for a fixed time, t, Z (.omega.,t) is a random variable. For a given .omega., Z(.omega.,t), as a function of time, t, is called a sample function or realization. Thus, a time series is a realization or sample function from a certain stochastic process. Typically, however, the variable .omega. is suppressed, and the process is written Z(t) or Z.sub.t. The process is then called a real-valued process because it assumes only real values. The imaginary value, .omega., is not treated. Moreover, for any given real-valued process {Z(.omega.,t): t=0,.+-.1,.+-.2, . . . }, the mean function of the process is given by .mu..sub.t =e(Z.sub..function.t), which may be used by the statistical collapser 204 to calculate the mean for all metering records collected by the resource manager 202.

Detailed Description Text (17):
The computing platform also includes a database 206 that stores data. In one embodiment, the database 206 stores the time series representing a performance metric as though its associated performance data had been collected at fifteen-minute intervals. That is, after the resource manager 202 collects the performance data from the computers 106 and after the statistical collapser 204 generates the time series representing performance data collected at fifteen-minute intervals, the database 206 stores the time series.

Detailed Description Text (18):
The database 206, in one embodiment, is capable of storing at least sixty gigabytes of performance data and can process at least one record per second. For example, the database 206 stores thousands of mainframe computer performance statistical descriptors and resource utilization data. Although the database 206 is depicted as a single database, according to the exemplary embodiment, the database 206 may be a plurality of databases. A database suitable for implementing the database 206 is a MICS database available from Computer Associates located in Santa Clara, Calif., or an SAS IT Service Vision database available from SAS Institute located in Cary, N.C.

Detailed Description Text (22):
It must be noted that if a time series contains too few time points, the time series may not be representative of the particular data under analysis, including, but not limited to Internet metering data, marketing data on the success or failure of product offerings, telephone usage patterns, cash flow analyses, financial data, customer survey data on product reliability, customer survey data on product preference, etc. For example, if the time series in the above example embodiment contains too few time points, the time series may not be representative of actual resource performance. That is, peak usages (or spikes) may not be detected if too few time points are taken. Therefore, sampling intervals which exclude such peaks may inaccurately represent the resource utilization. Thus, the number of time

points in the time series may be determined by the particular use of the performance metric, the particular resource, the particular computing platform, etc. To illustrate, suppose a network manager that is responsible for monitoring the behavior and effectiveness of the computing platform 100 resources monitors the performance and activities for each computer 106. The system network manager tracks the computing platform 100 resources performances by gathering the appropriate performance data from each component or network element in the computing platform 100. As described, performance metrics to be monitored include, but are not limited to, CPU consumption percentage, disk drive usage percentage, Internet traffic, users logged on to the Internet, network communication packet traffic, and users logged on to a particular server computer, for example.

Detailed Description Text (23):
Suppose further that the computing network 102 typically includes entities such as a configuration management team and a performance management team. The configuration management team would plan the computing platform 100's growth and modernization. Accordingly, weekly data points would be adequate for these network planning purposes. Daily data points may be more appropriate for use by the performance management team, however. This is because the performance management team would be concerned with maintaining the computing platform 100's error-free performance. Accordingly, the performance management team would be concerned about peak usages (or spikes) in resource consumption of the computing platform 100. Monitoring spikes in the computing platform 100 would facilitate load sharing, for example.

Detailed Description Text (24):
Referring to FIG. 3, one embodiment of the present invention generates accurate CPU utilization descriptors in the following manner. The resource manager 202 for the computer 106 collects the performance data 301 and provides it in the form of one-second metering records 302 to the statistical collapser 204. The statistical collapser 204 statistically collapses the one-second records 302 into fifteen-minute data, which is stored in a file of performance data 304 of the database 206. The data extractor 208 extracts the performance data 304 from the database 206 and provides it to the statistical collapser 210. The statistical collapser 210 statistically collapses the fifteen-minute data into one-week data.

Detailed Description Text (27):
It must be noted that few time series are statistically stationary. The computing platform 100 thus includes a time series analyzer 212 to determine whether the time series generated by the statistical collapser 210 is statistically stationary. According to one embodiment, the time series analyzer 212 analyzes probability values for a plurality of X.sup.2 (chi-square) tests to make this determination. "Chi-square tests" as used herein generally define generalizations and extensions of a test for significant differences between a binomial population and a polynomial population, wherein each observation may fall into one of several classes and which furnishes a comparison among several samples rather than just between two samples. Such chi-square tests include a test of residuals, tests of hypotheses, tests of significance, tests of homogeneity, tests of association, goodness of fit tests, etc., as is known in the relevant art. In the embodiment where the resource is computer 106 and the performance metric is CPU utilization, the time series analyzer 212 determines whether there is a statistically significant correlation between a particular CPU utilization value and the value for CPU utilization for the previous time period by reviewing correlation and covariance statistics. Tables 1-4 list chi-square values for a test for residuals for the computers 106a-106d, respectively. The column "DF" refers to the degrees of freedom of variation among a set of scores. In particular, column "DF" refers to the degrees of freedom of variation among a set of metering records for CPU utilization. To illustrate, suppose there is a set of ten scores. Statistically the degrees of freedom given by

Detailed Description Text (40):
II. Autoregressive Forecasting of Computing Resources

Detailed Description Text (41):
FIG. 4 depicts a flow chart of a collecting, collapsing, and regresssing process 400 suitable for use in one embodiment of the present invention. Task 402 starts the process 400, where control immediately passes to task 404. Task 404 extracts performance data from at least one of the computers 106. In one embodiment, the resource manager 202 extracts the performance data from its associated computer every second.

Detailed Description Text (45):
Task 412 applies the autoregressive modeling tool 216 to the time series to generate forecasts of the computing platform 100 resources. Task 412 also generates recursive forecasts whereby actual future performance data is fed back into the autoregressive modeling tool 216 to calibrate the system and method. Task 414 completes the process 400. This process provides a turnkey solution to CPU utilization forecasting that can be implemented easily by any system network manager.

Detailed Description Text (46):
Referring back to FIG. 1, the computing platform 100 may include a results processor 104. The results processor 104 generates graphical representations of performance data extracted from the computing platform 100. The results processor 104 generates information for use in written reports that document the results of the process 400. In the embodiment where the performance metric is resource utilization and the resource is a central processing unit of the computers 106, the results processor 104 produces reports and graphical representations of comparisons of actual CPU utilization with CPU utilization forecasts.

Detailed Description Text (55):
These and other changes can be made to the invention in light of the above-detailed description. In general, in the following claims, the terms used should not be construed to limit the invention to the specific embodiments disclosed in the specification and claims, but should be construed to include all computer platforms that operate under the claims to provide a system and method for computing resource forecasting utilization.

CLAIMS:

1. In a computing platform having a plurality of resources, each resource includes at least one aspect, a method for forecasting at least one aspect of the plurality of resources to produce computing platform resource performance forecasts, the method comprising the steps of: collecting at intervals performance data associated with a performance metric from a computing platform resource; statistically collapsing the collected performance data associated with a performance metric at least once to produce a time series and storing the time series; determining whether the time series is statistically stationary using a plurality of chi-square tests; converting the time series to a statistically stationary time series when the time series is determined not to be statistically stationary; bypassing the converting when the time series is determined to be statistically stationary; adding a date/time stamp to the time series, including converting the time series date/time stamp to a number of seconds equivalent to the value represented by the date/time stamp; applying an autoregressive modeling tool to the time series to produce computing platform resource performance forecasts; and outputting the resource performance forecasts in a format such that computing platform resources can be modified based on the data.

2. A system to forecast performance of at least one computing platform resource, comprising: a resource manager for collecting performance data associated with a

performance metric from a computing platform resource; a first statistical collapser receiving input from the resource manager, the first statistical collapser collapsing the data received from the resource manager into a first time series; a second statistical collapser, the second statistical collapser collapsing the data received from the first statistical collapser into a second time series; a time series analyzer coupled to the second statistical collapser and configured to: determine whether the second time series is statistically stationary using a plurality of chi-square tests, convert the second time series to a statistically stationary time series when the second time series is determined not to be statistically stationary, and bypass the converting when the second time series is determined to be statistically stationary; a time point converter coupled to the time series analyzer, the time point converter being operable to add a date/time stamp to the second time series and wherein the time point converter converts the time series date/time stamp to a number of seconds equivalent to the value represented by the date/time stamp; an autoregressive modeling tool coupled to the time series analyzer, the autoregressive modeling tool producing computing platform resource performance forecasts; and a result processor coupled to the autoregressive modeling tool.

6. The system according to claim 2, wherein the performance metric represents utilization and the computing platform resource comprises a central processing unit.